

# **MIST-IFYING MSP AUTOMATION**

**MSP networking adventure**

**Steinn Örvar Bjarnarson 2023**

# Who is this guy?

- **Steinn Bjarnarson (Steinzi)**
  - Lead product development engineer at Advania
  - Network engineer turned Automation specialist/developer
  - The opinions expressed are neither mine nor my employers.



**Can we automate our data centers and customers' enterprise networks using the same tools?**

**Spoiler: YES**

# Overview

## outline of the talk

MSP network automation

DC vs. Enterprise environments

Declarative vs Imperative IaC

Apstra + Mist automation

Streamlining our network services

Testing CI/CD pipelines

Tools, tips, and tricks!

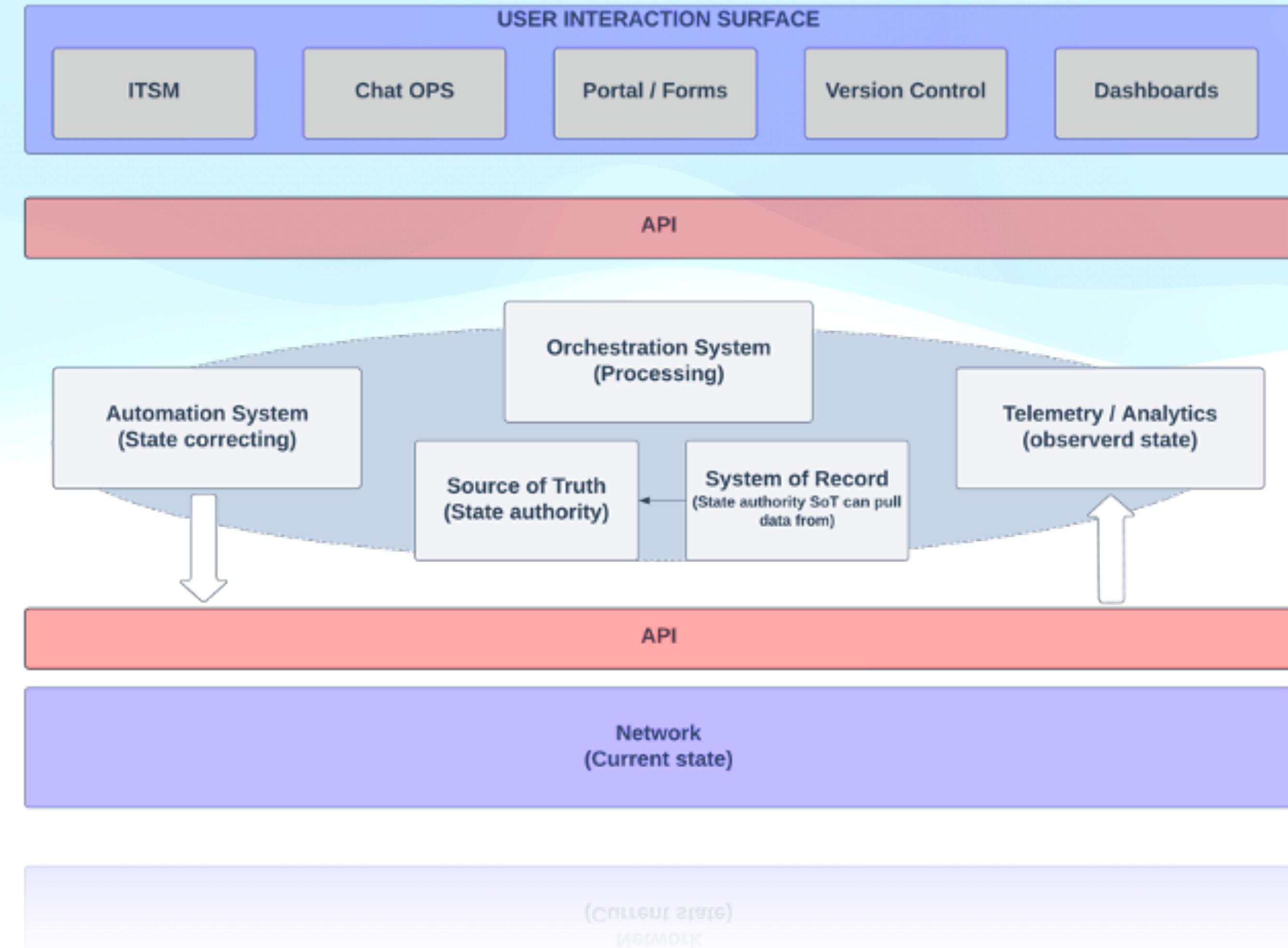
# MSP network automation

(For enterprise customers)

# Network Automation

## Big-picture

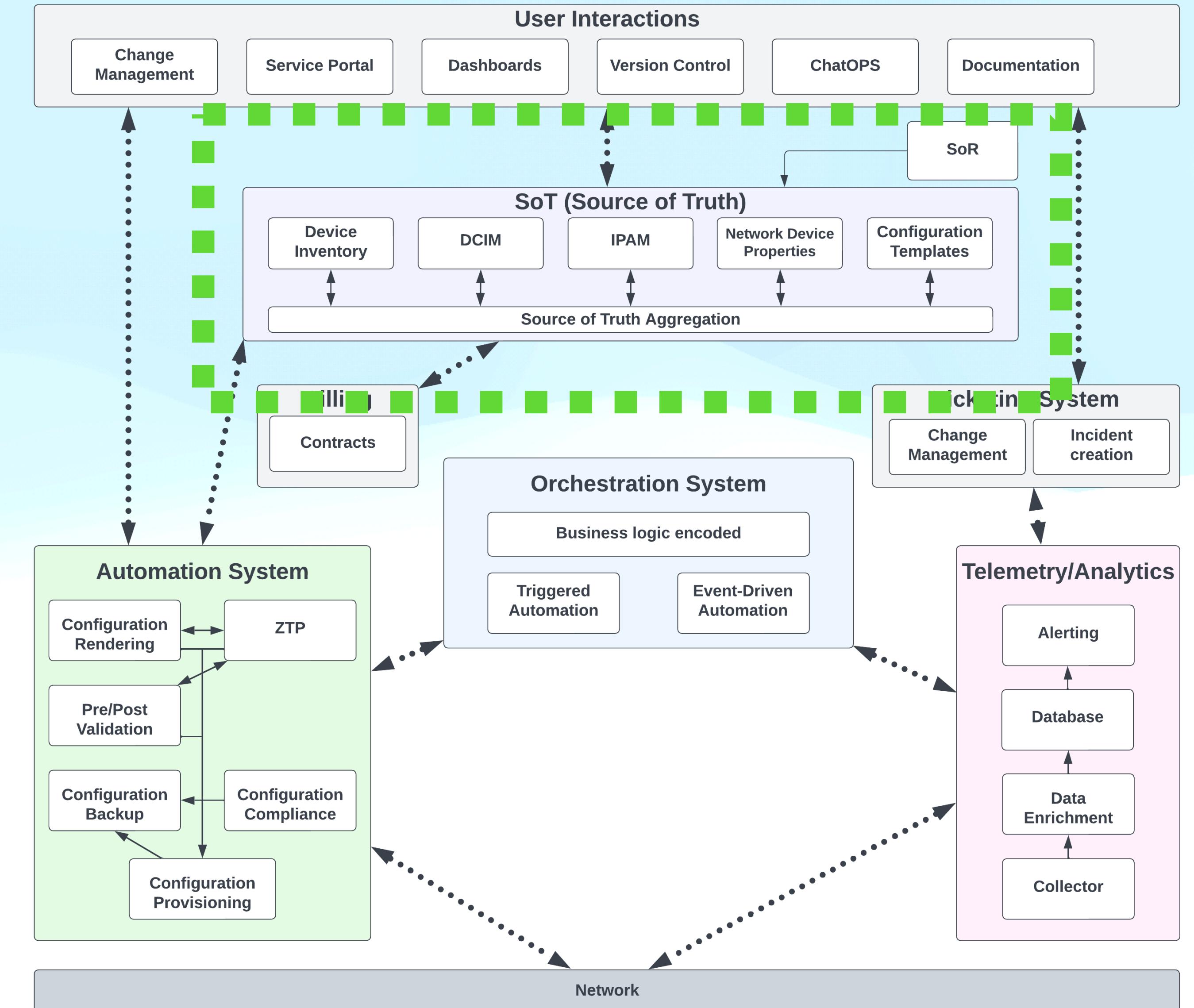
- Source of truth (SoT) 
- System of record (SoR)
- Orchestration/Automation system 
- Telemetry & Monitoring



# Source of Truth

## Heart of the system

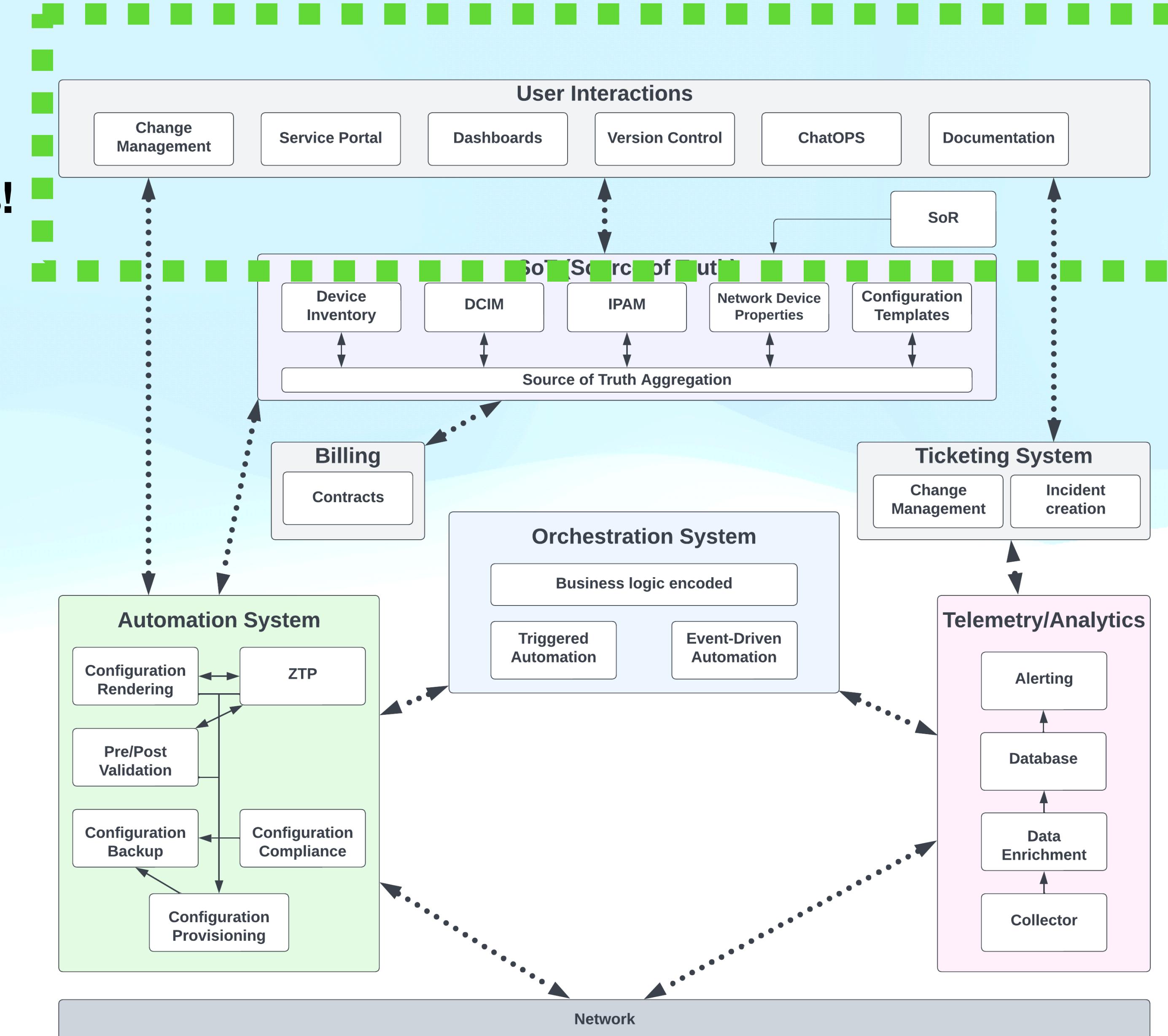
- Holiest of holies
- Netbox/Nautobot/DB
- SoR integration is the tricky part
- What is an SoR?



# User interaction

Please keep your hands off my networks!

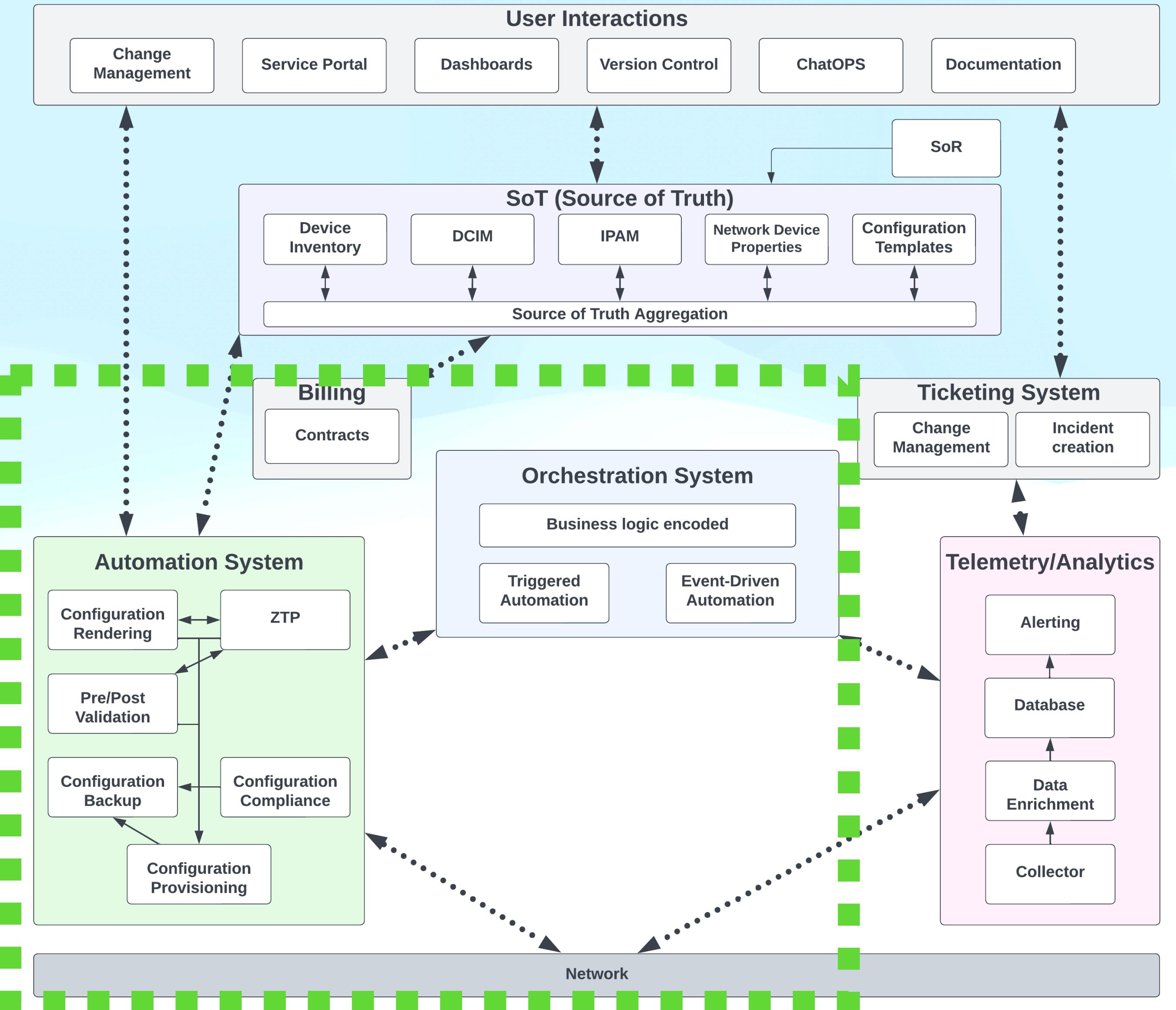
- Abstract business logic and processes into other systems
- Consumable parts like Documentation
- Good change management is critical!



# Automation/Orchestration

## where the magic happens

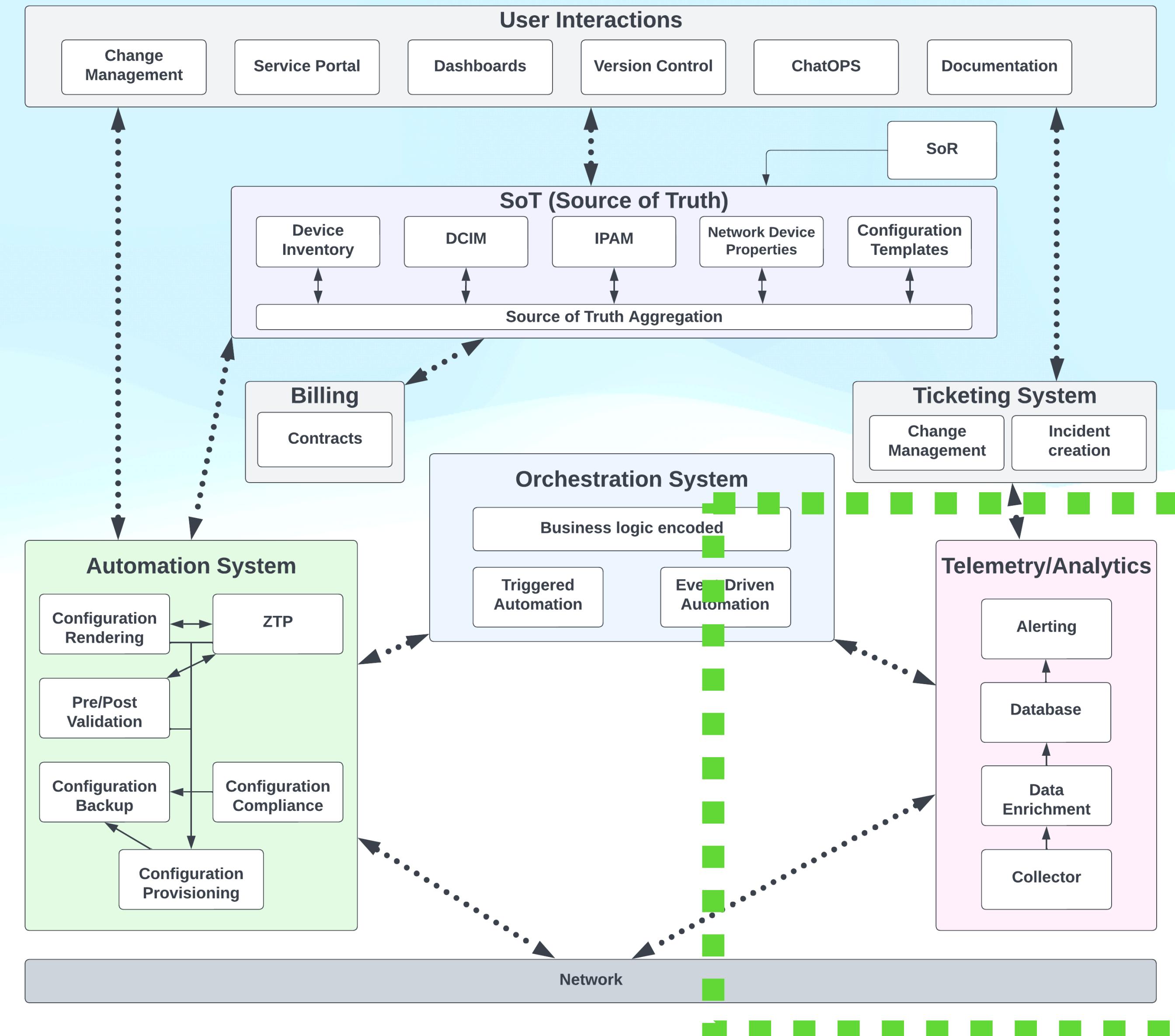
- Lots of options for this part
- More on this later!
- Watch out for vendors selling you solutions that handle everything here!



# Telemetry/Analytics

**Alert! Alert!**

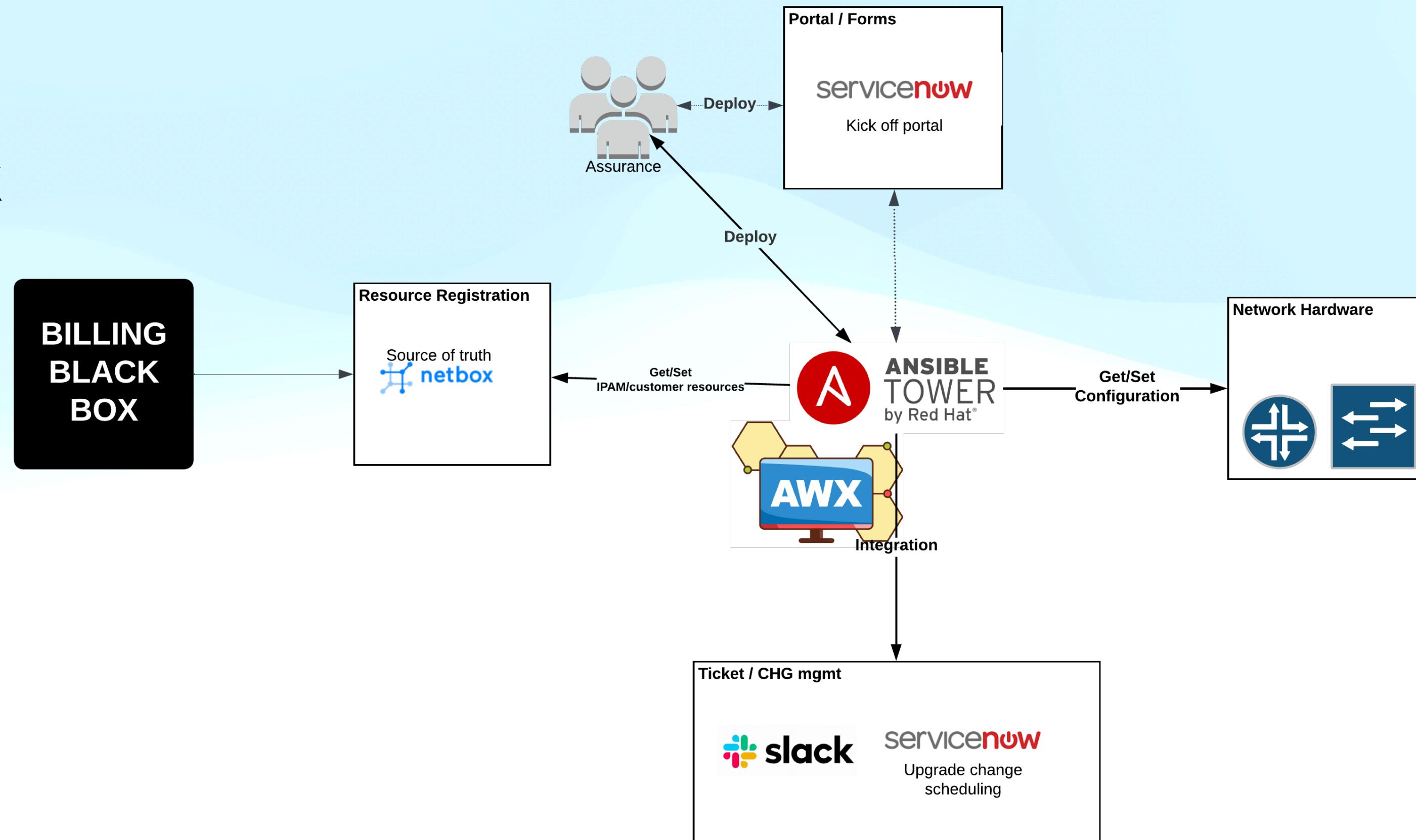
- Incident creation
- Monitoring of our systems



# Why go through all this?

## Benefits of this design

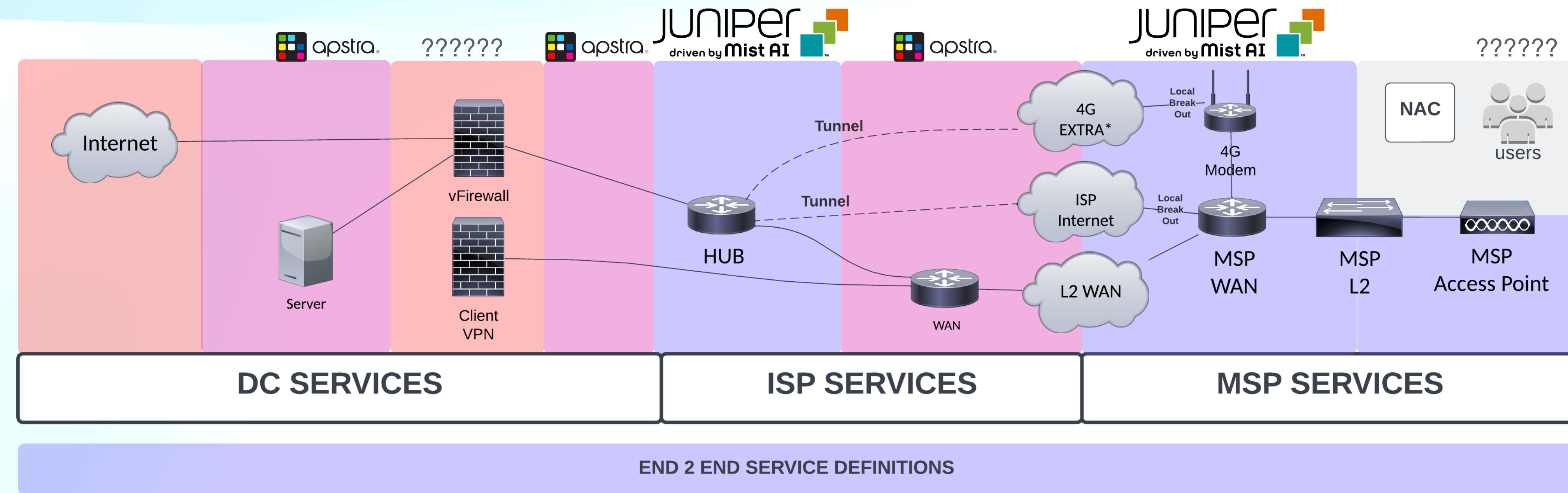
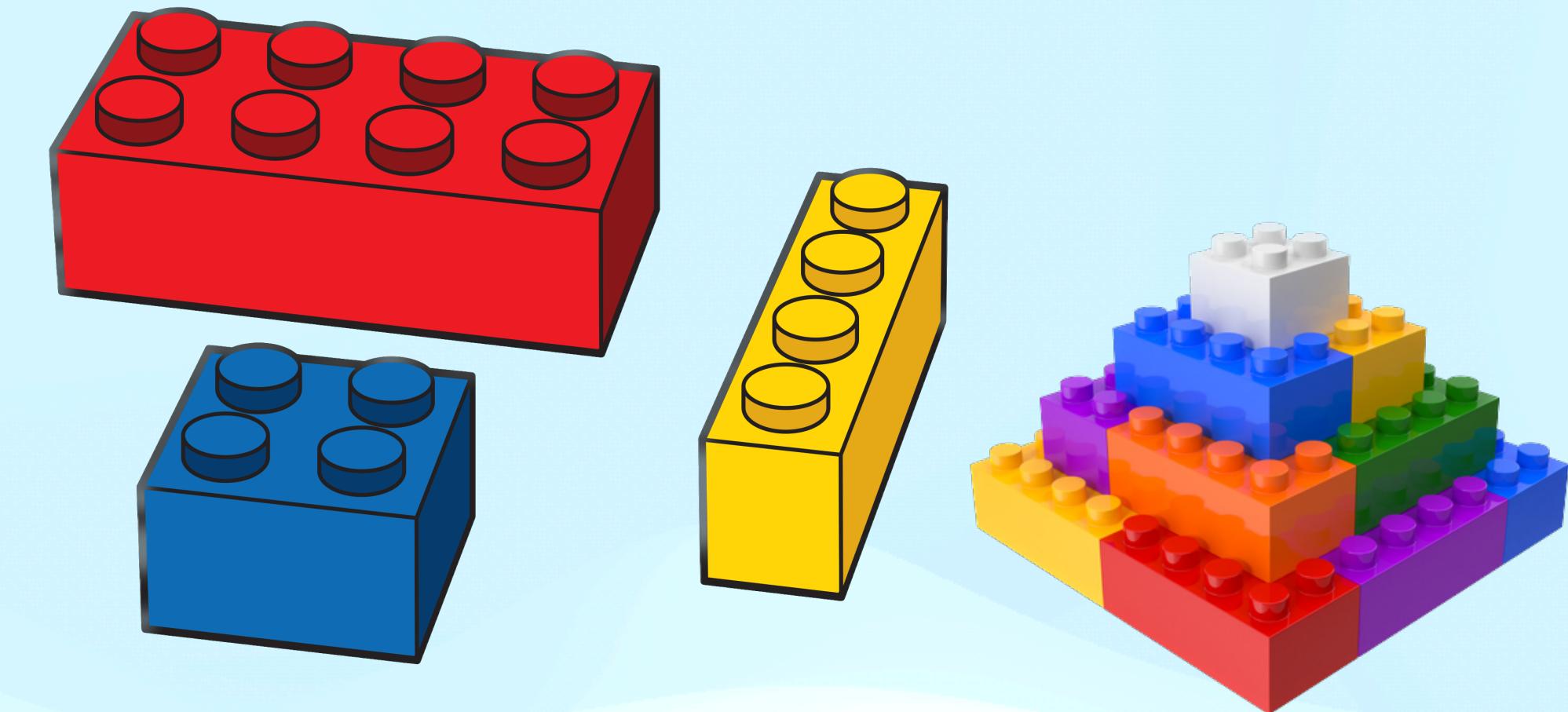
- Separation of billing and network vendors & services
- Authoritative state source (SoT)
- Multivendor support (Sorry, Marvis)
- Sensible deployment defaults
- Reduced human error
- End 2 end service definitions?



# Network Service Orchestration

## “Lego orchestration”

- End 2 end service chaining/definitions
- Service defined from DC/Cloud to the user
- Billing and resource mapping to customers



# How do we get started?

# **Let's have Pasta**

**Imperative Step-by-step (HOW)**

**VS.**

**Declarative: The desired result (WHAT)**

# Imperative (HOW) Step-by-step Recipe

- We focus on the HOW, and we should get WHAT we want
- The recipe lists each step in detail
- Highly coupled in most cases
- We get pasta



```
begin
    fill_pot_with_water()
    place_pot_on_stove()
    turn_stove_on("high")
    wait_for_water_to_boil()

    add_salt_to_boiling_water()
    open_pasta_box()
    add_pasta_to_boiling_water("spaghetti")
    set_timer("al dente")
    stir_pasta_periodically()

    while timer_is_running:
        chop_onion()
        chop_garlic()
        heat_pan_on_stove("medium")
        add_olive_oil_to_pan()
        saute_onion_and_garlic()

        open_canned_tomatoes()
        add_tomatoes_to_pan()
        add_seasonings("basil", "oregano", "salt", "pepper")
        simmer_sauce()

        prepare_meatball_mixture()
        shape_meatballs()
        heat_another_pan_on_stove("medium")
        add_olive_oil_to_pan()
        cook_meatballs_in_pan()

    stop_timer()
    drain_pasta_in_colander()

    add_cooked_pasta_to_sauce()
    mix_pasta_and_sauce()
    add_cooked_meatballs_to_pasta()
    grate_parmesan_cheese()
    sprinkle_cheese_on_top()

    serve_pasta_on_plate()
end
```

# Declarative (WHAT)

## Desired end result

- We order **WHAT** we want by declaring it to the waiter.
- Focus only on outcomes.
- I don't care how you do it.
- We get pasta.
- What about the cooks in the kitchen?



[RODNAE Productions](#)

```
1  order:  
2    dish: pasta  
3    pasta_type: spaghetti  
4    sauce: marinara  
5    extras:  
6      - meatballs  
7      - parmesan_cheese  
8    al_dente: True  
9
```

# Sudo code comparison

## Imperative

```
begin
    fill_pot_with_water()
    place_pot_on_stove()
    turn_stove_on("high")
    wait_for_water_to_boil()

    add_salt_to_boiling_water()
    open_pasta_box()
    add_pasta_to_boiling_water("spaghetti")
    set_timer("al dente")
    stir_pasta_periodically()

    while timer_is_running:
        chop_onion()
        chop_garlic()
        heat_pan_on_stove("medium")
        add_olive_oil_to_pan()
        saute_onion_and_garlic()

        open_canned_tomatoes()
        add_tomatoes_to_pan()
        add_seasonings("basil", "oregano", "salt", "pepper")
        simmer_sauce()

        prepare_meatball_mixture()
        shape_meatballs()
        heat_another_pan_on_stove("medium")
        add_olive_oil_to_pan()
        cook_meatballs_in_pan()

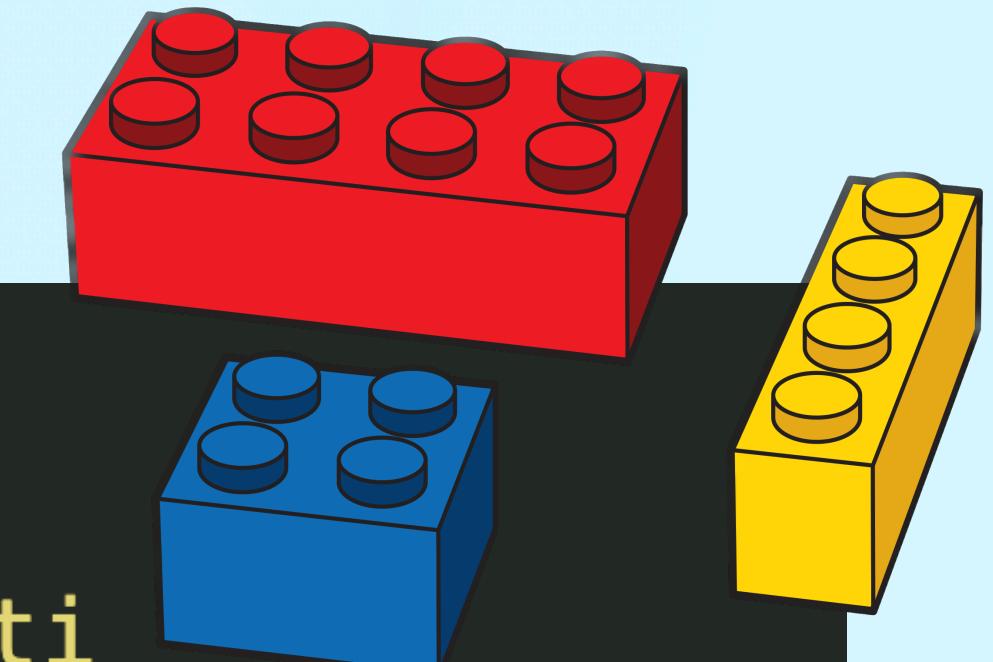
    stop_timer()
    drain_pasta_in_colander()

    add_cooked_pasta_to_sauce()
    mix_pasta_and_sauce()
    add_cooked_meatballs_to_pasta()
```



## Declarative

```
1   order:
2       dish: pasta
3       pasta_type: spaghetti
4       sauce: marinara
5       extras:
6           - meatballs
7           - parmesan_cheese
8   al_dente: True
9
```



COMPLEX UNDER THE HOOD  
(EASY TO USE)

# Automation abstraction level

## What does my automation target?

- Try to go as high as we can
- This depends on a lot of factors
- No one-size-fits-all
- Build your plan from the bottom up

Customer

Site

Switches

Switch

Port

# Automation in the DC vs. Enterprise

# DC vs Enterprise

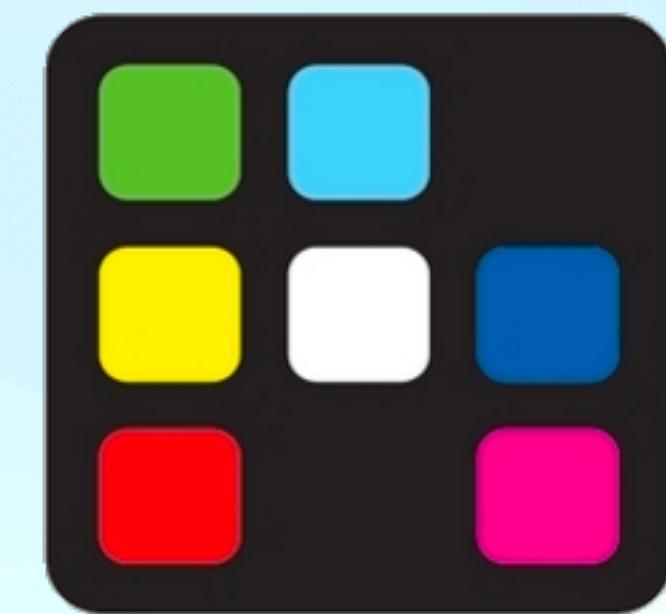
## Heterogeneous versus non-heterogeneous

- The relatively static physical environment
- Standardized deployment procedure for everything that can happen
- Complete control of infrastructure equipment vendor.
- Physical gear is only accessible by authorized personnel
- Dynamic with unexpected occurrences
- Mixed bag deployments, ad-hoc integrations
- Partial/Full/no governance over network equipment
- Unknown users have physical access to some network equipment.

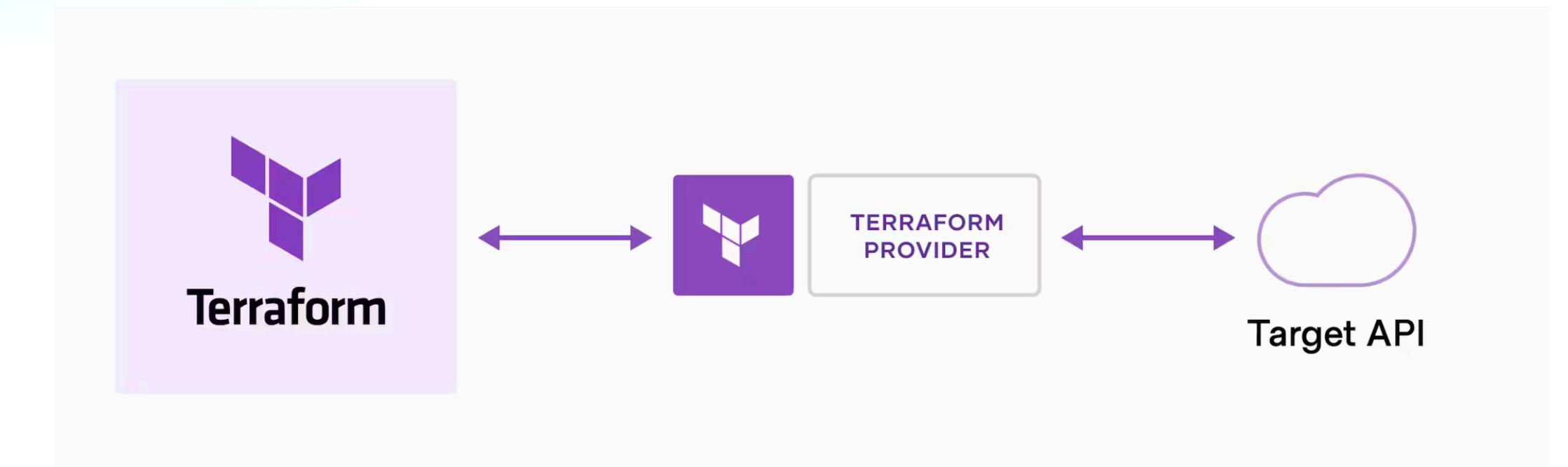
# DC side

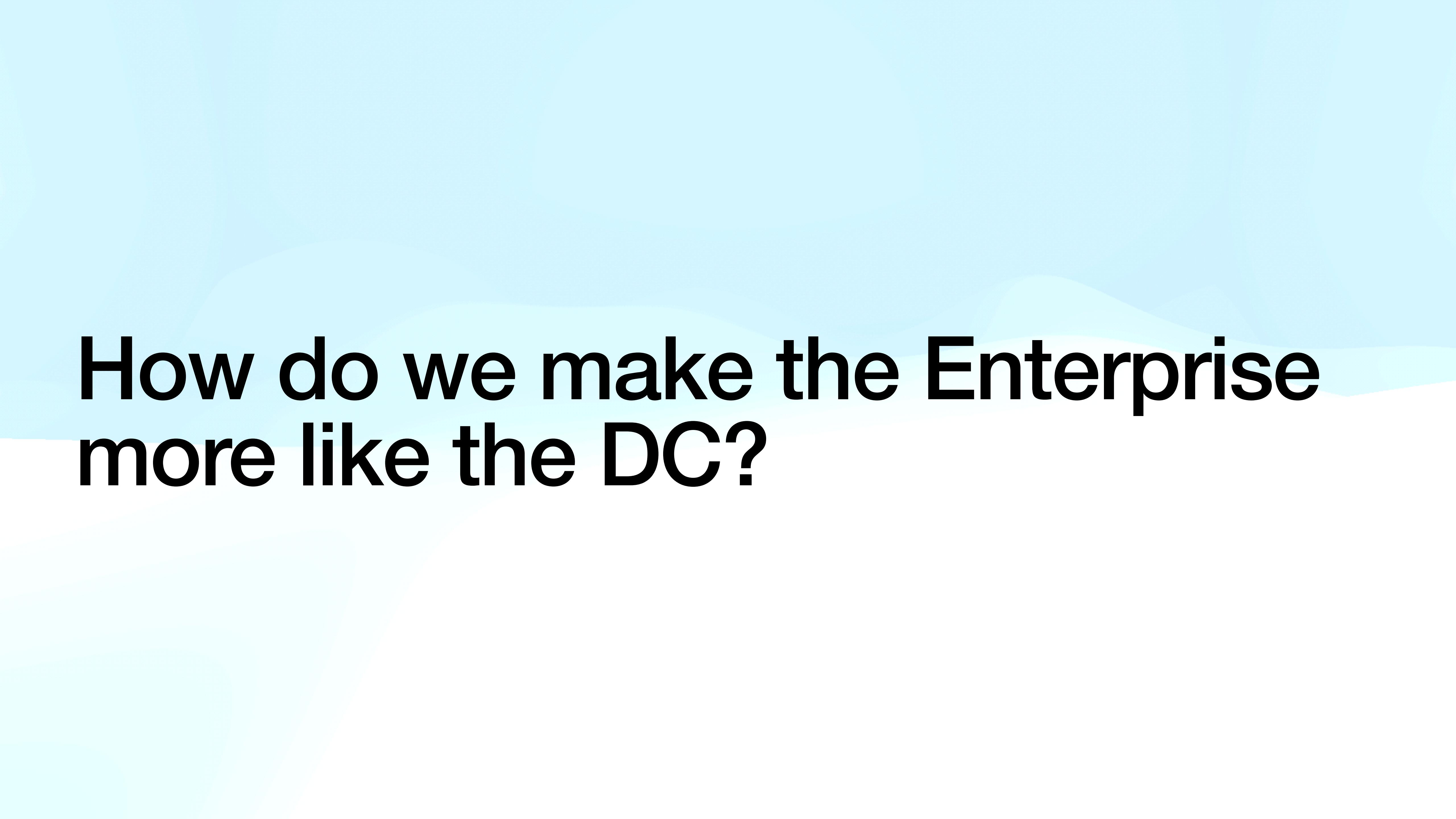
## Juniper Apstra + Terraform

- Abstracts the implementation details
- Declare your intended state of the network
- 1st party support for Terraform from Juniper



apstra®

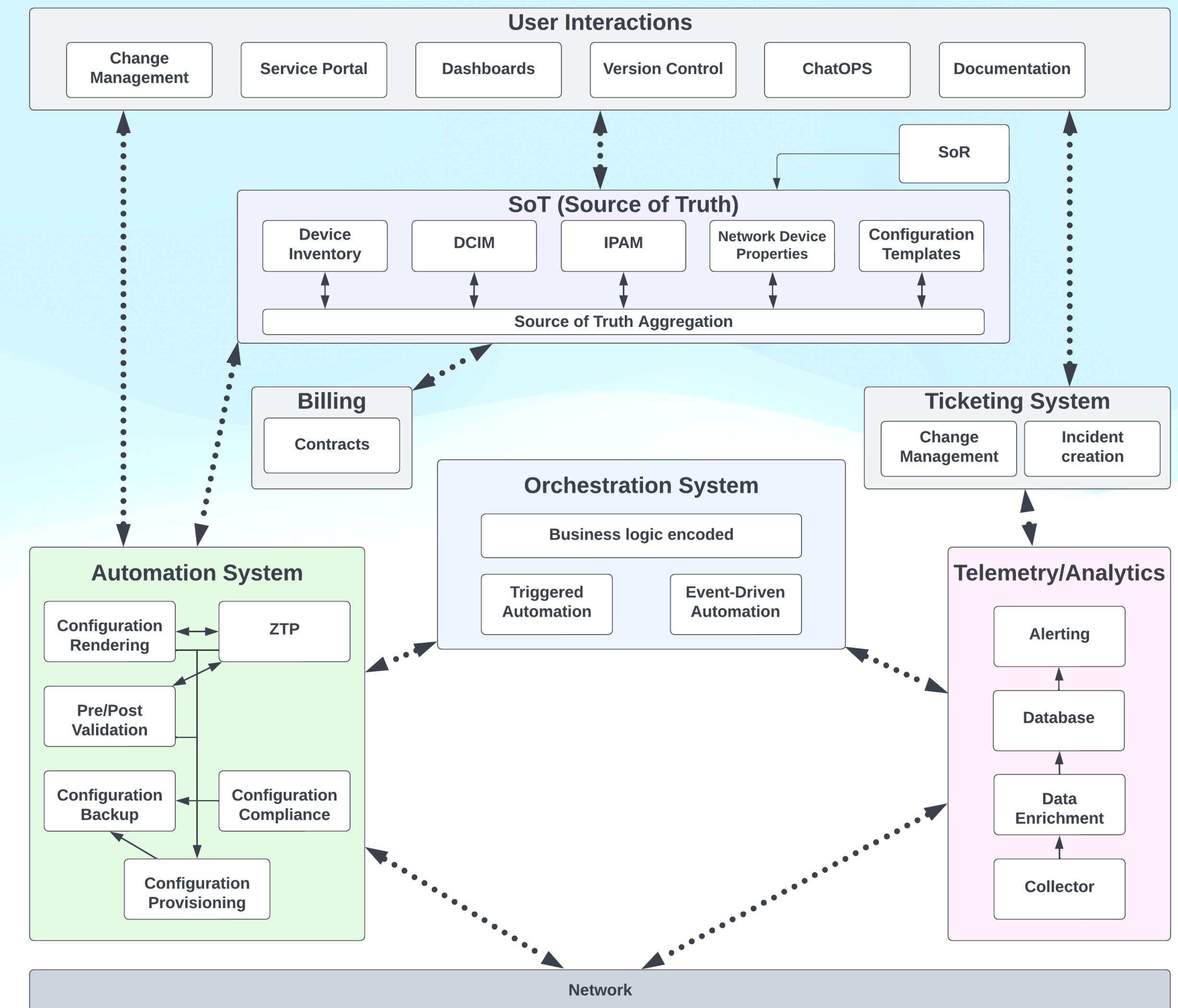




**How do we make the Enterprise  
more like the DC?**

# Enterprise side Enterprise abstraction

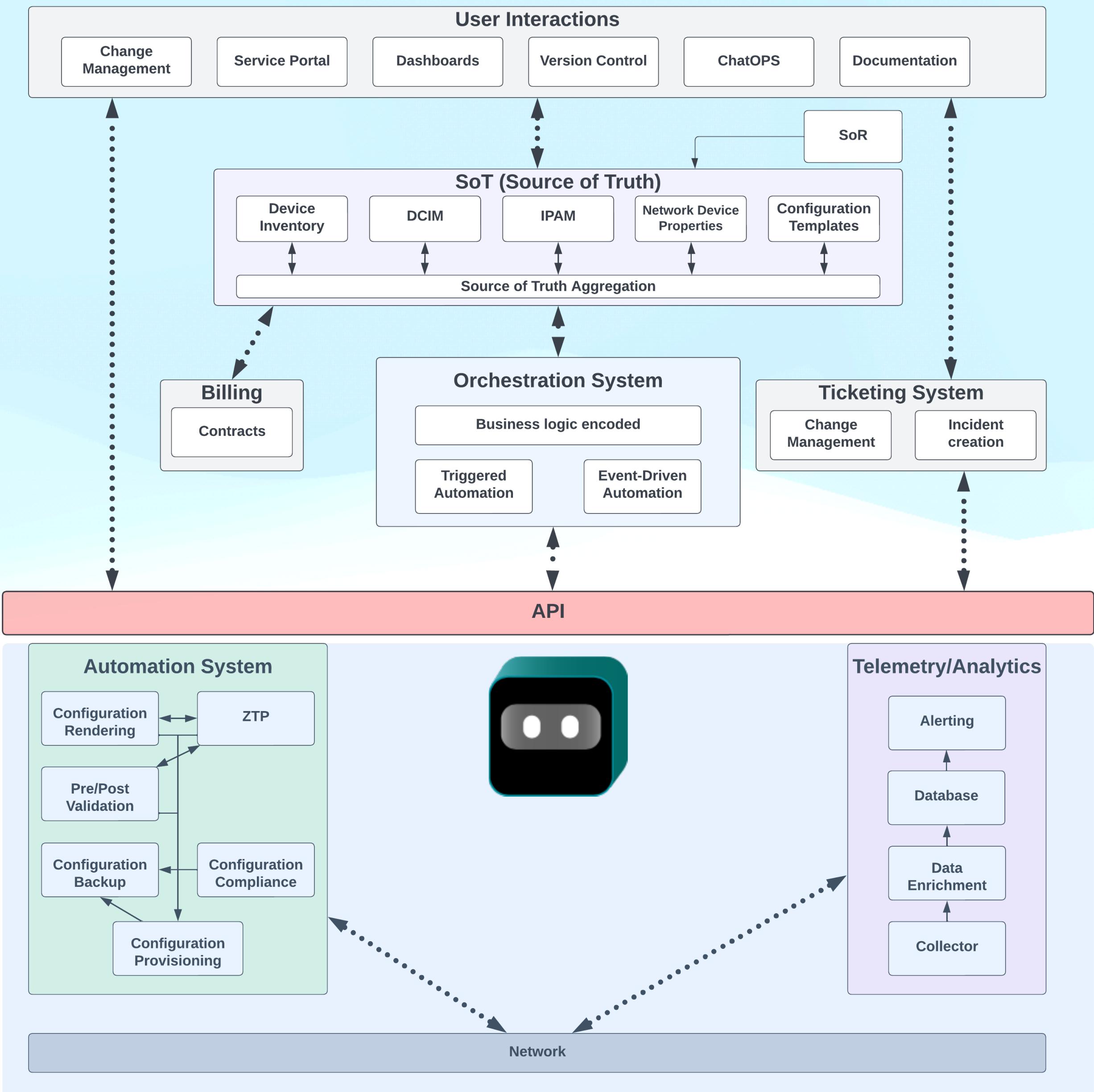
- We need something sitting between “us” and the network
- Stop thinking about specific switches and APs and think of the whole thing as services.
- We want cattle, not pets.



# Mist & Marvis

## Abstract away the devices

- Full access to everything via API
- Template ORG's
- **SITE VARIABLES!**
- WAN / Switch / AP / RF Templates
- Dynamic Port Configuration (DPC)



# Templates/Template orgs

## UI / API hybrid work!

- Create template ORGs in your MSP portal!
- Let your domain expert work inside the UI on the templates!
- Handle new features & API changes.
- Copy paste!

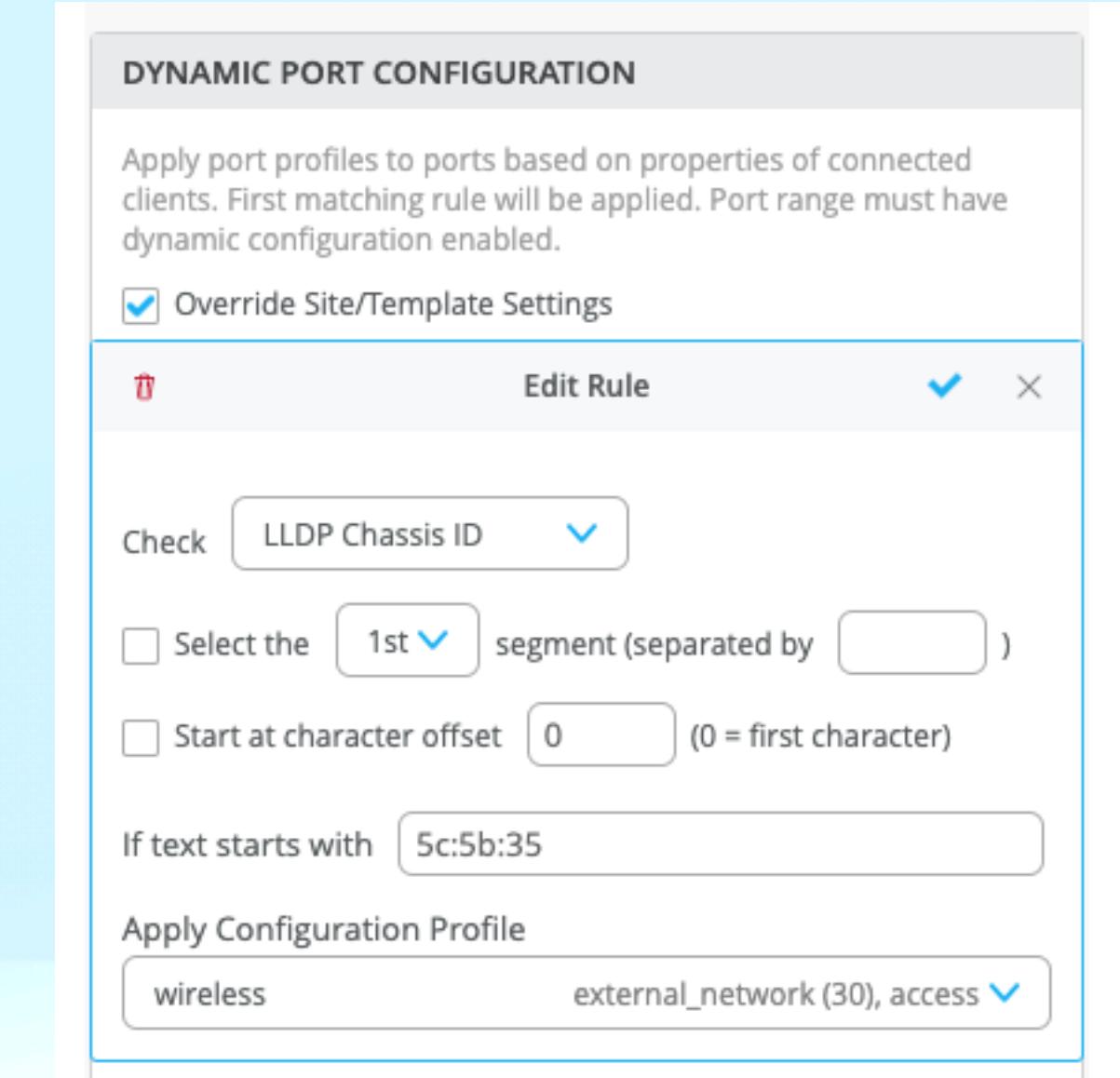
The screenshot displays three panels of a network configuration interface:

- Site Variables:** A table showing site-level variables and their values. The variables listed are {{BGP\_LOCAL\_AS}}, {{DNS\_SERVER\_PUBLIC\_01}}, {{DNS\_SERVER\_PUBLIC\_02}}, {{DNS\_SERVER\_PRIVATE\_01}}, {{DNS\_SERVER\_PRIVATE\_02}}, {{PRIVATE\_WAN\_IP}}, {{PRIVATE\_WAN\_PREFIX}}, {{PRIVATE\_WAN\_GW}}, and {{BRANCH\_MGMT\_NETWORK}}. Their corresponding values are 65501, 8.8.8.8, 8.8.4.4, 8.8.8.8, 8.8.4.4, 169.254.0.1, 29, 169.254.0.6, and 10.250.3 respectively.
- WLANS:** A table showing wireless network configurations. It includes columns for SSID, Band, VLAN ID, and Security. The entries are: {{CUST\_NAME}} Guest (2.4GHz, 5GHz, VLAN 999, WPA2/PSK), {{CUST\_NAME}} IOT (2.4GHz, 5GHz, VLAN 20, WPA2/PSK), and {{CUST\_NAME}} (2.4GHz, 5GHz, VLAN 10, WPA2/EAP (802.1X)).
- VLAN:** A configuration panel for setting up VLANs. It includes options for Untagged, Tagged (which is selected), Pool, and Dynamic. A text input field for "VLAN ID" contains the placeholder "{{VLAN\_VAR}} (1 - 4094)".

# Dynamic port Configuration

## A config, not a security feature!

- Let's not worry about each switch and focus on the service
- It's a configuration and deployment feature, not a security feature.



```
        "port_usages": {
            "dynamic": {
                "mode": "dynamic",
                "rules": [
                    {
                        "src": "lldp_system_description",
                        "usage": "uplink",
                        "equals": "Juniper",
                        "expression": "[0:7]",
                        "hasOffset": false,
                        "offset": "0",
                        "__id": "c7e2eba6-c560-418f-bc94-f783949354e9"
                    },
                    {
                        "src": "lldp_system_description",
                        "usage": "ap",
                        "equals": "Mist",
                        "expression": "[0:4]",
                        "hasOffset": false,
                        "offset": "0",
                        "__id": "48fb434c-ae17-43d7-b67b-c6ae71011a92"
                    }
                ]
            }
        }
```

# Enterprise side (backend)

## Juniper Mist + Ansible

- We use Mist features to get closer to intent-based networking.
- Focusing on the WHAT, not the how.
- No need for a separate inconsistency detection solution



```
- name: Assign site templates
  hosts: "STATIC_ORG_ID"
  gather_facts: false
  vars_files:
    - msp_api_token.vault
  tasks:
    - name: Org declaration using template org
      mist_org:
        tenant_slug: NASA
        template_org_id: aabf52b5-9cf3-4444-9f15-372478c22222
        msp_id: a8061fc2-1234-4473-8f7d-ava111bbb6eb
        api_token: "your_api_token"
        mist_api_url: https://api.eu.mist.com
        state: present
    - name: Site declaration
      mist_site:
        org_id: "your_org_id_or_name"
        site_id: "your_site_id_or_name"
        site_location: "your_site_location"
        site_lat: "your_site_lat"
        site_lon: "your_site_lon"
        api_token: "your_api_token"
        mist_api_url: https://api.eu.mist.com
        vars:
          site_var1 : "site_var1_value"
    - name: Site template declaration
      mist_site_template:
        org_id: "your_org_id_or_name"
        site_id: "your_site_id_or_name"
        api_token: "your_api_token"
        mist_api_url: https://api.eu.mist.com
        rftemplate_id: "your_rf_template_id_or_name"
        networktemplate_id: "your_network_template_id_or_name"
        gatewaytemplate_id: "your_gateway_template_id_or_name"
```

# Enterprise side (frontend)

## Juniper Mist + Ansible

- Simplified options
- deliver custom automation on top of MIST
- Declarative & Idempotent
- SE can configure
- Then just ship devices!  
(After the wifi survey...)

Launch | MIST - new deployment

1 Survey      2 Preview

Customer ID \* ?  
aaa

Site ID \* ?  
aaa-123

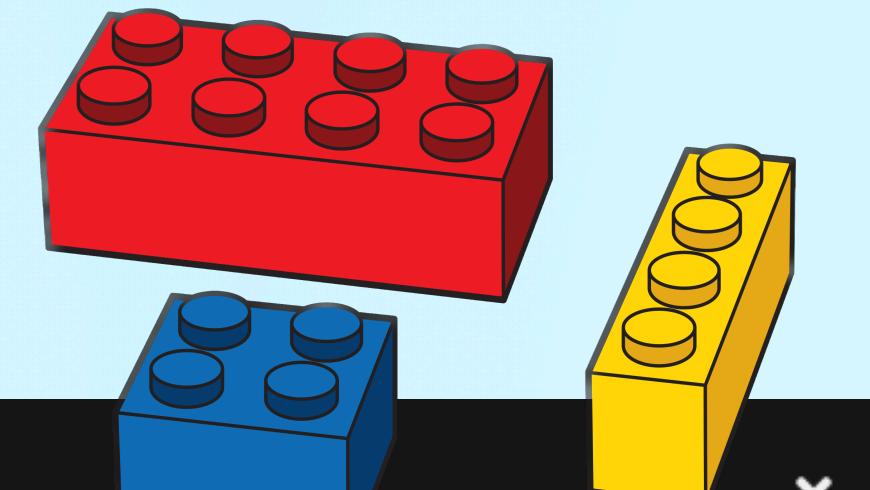
Automatic update day \* ?  
Thursday

Automatic Update Time \* ?  
03:00

SD-WAN \* ?  
true

enterprise network \* ?  
True

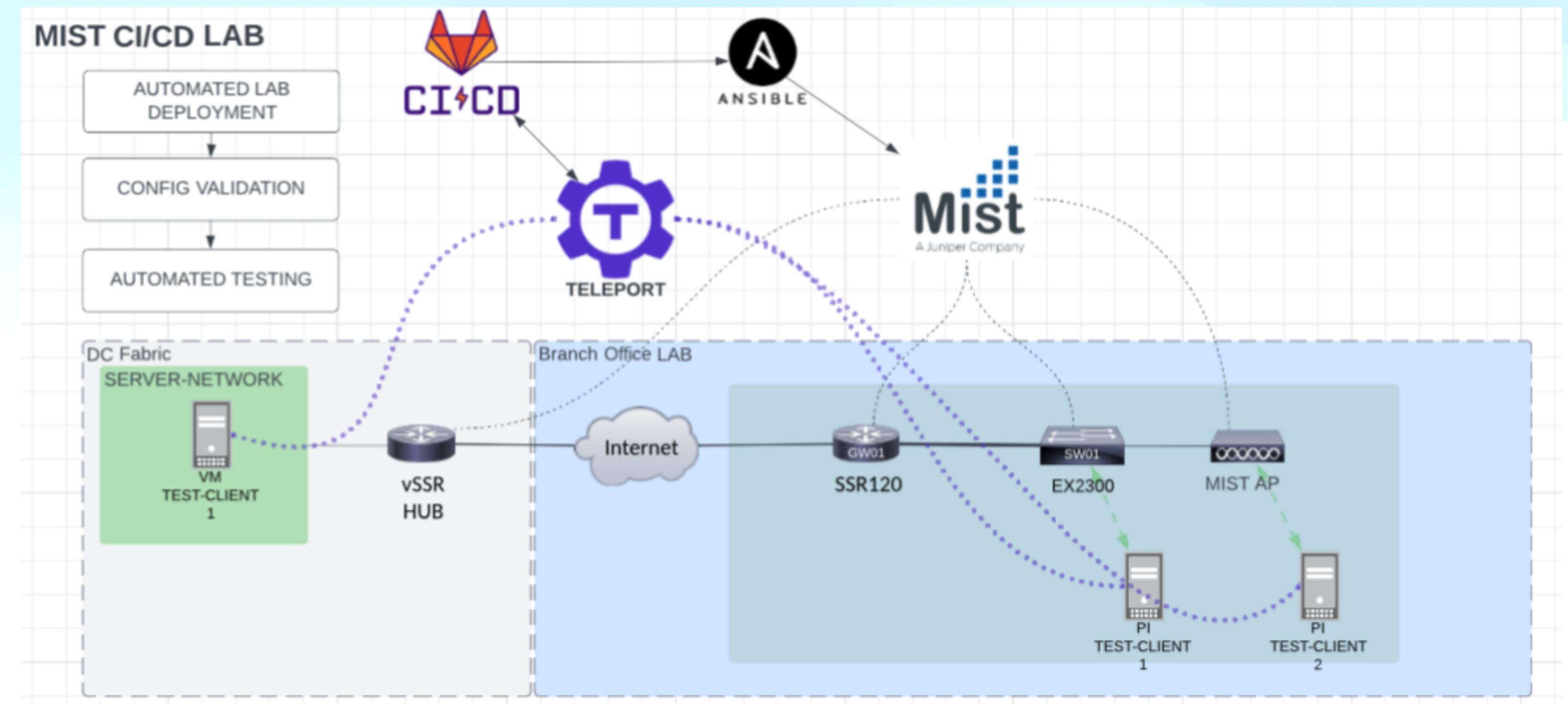
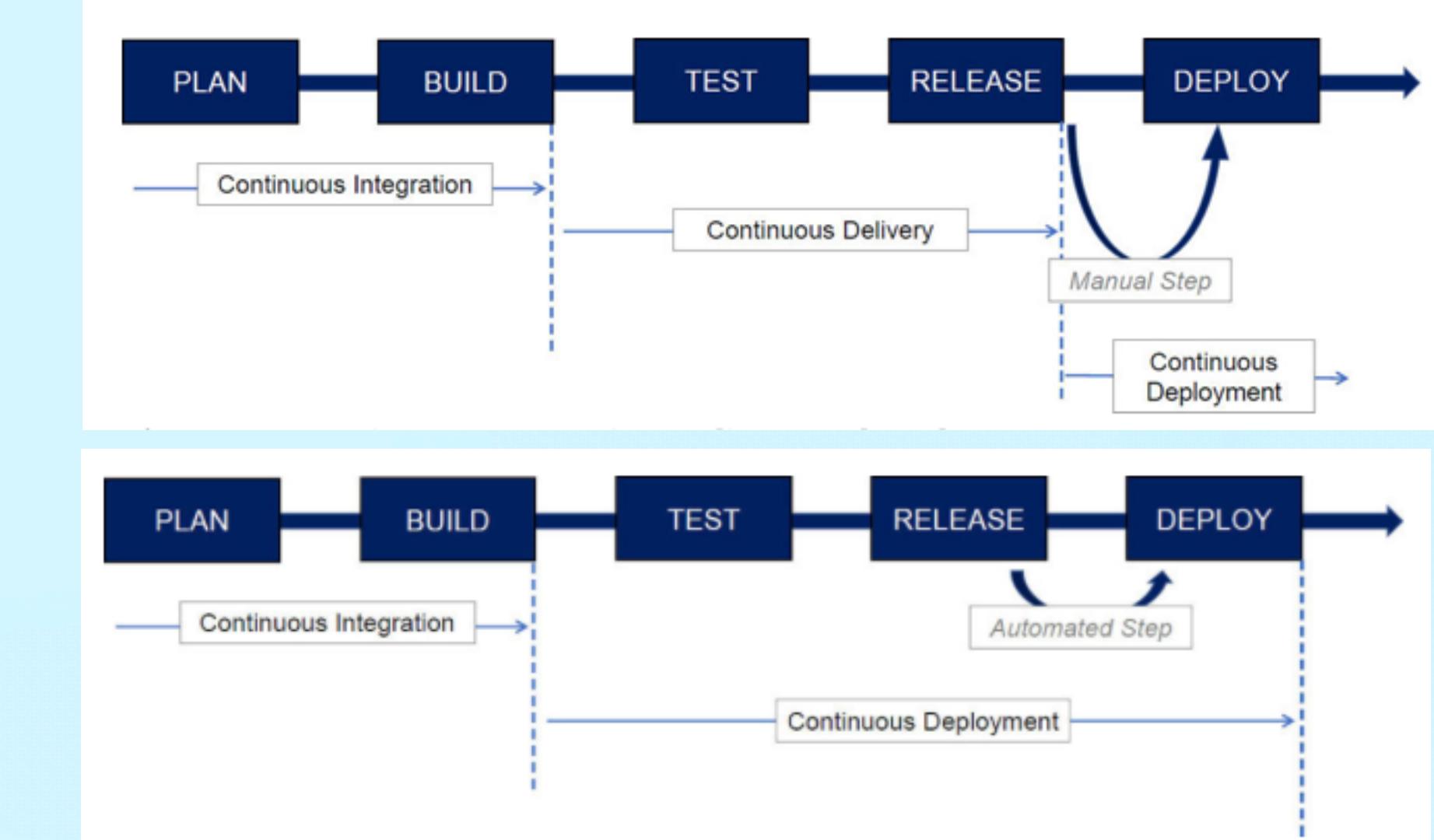
Cancel



# Testing

## CI/CD Template validation

- Automatic End2End service testing
- Tests
  - Internet access
  - Backend services
  - Wifi access



# Getting started

## Low hanging fruit

- Update automation
  - scheduling APs, then switches followed by WAN devices at a specific reoccurring date “Patch window.” defined in the customer contract
- Sensible defaults
  - deployment pipelines for new customers and adding new sites to existing customers.

# Tools

# APPS.MIST-LAB.FR

All kinds of neat stuff!

- From SNMP translation to app analytics



Mist LAB Demo Apps

Tags: alerting, configuration, example, monitoring, openapi, psk, reporting, rogue, script, snmp, switch, sync, tool, webapp, webhook, websocket

**App Analytics**  
Frontend angular, Backend django  
Web UI leveraging Mist APIs to generate reports about applications used. It can be used to view the applications used on a site or by a device, and to get the list of devices using an application.  
reporting, webapp  
Demo, Readme, Github

**Installer**  
Frontend angular, Backend django  
Web UI using Mist Installer APIs to provide a web interface to claim/edit/provision new devices. It can also be used by super admins and network admin to enable the "installer" access, allowing installer admins to get access to a specific site.  
configuration, webapp  
Demo, Readme, Github

**LDAP Sync**  
Script python  
Python script used to synchronise the Mist PPSK with AD/LDAP account. It is retrieving the list of PPSK configured for a specific SSID and the list of users in the Directory, and automatically create/delete the PPSKs.  
psk, script, sync  
Readme, Github

**Mist Want To Talk**  
Backend flask  
Mist Want To Talk is a Python app using Flask server to listen for Mist webhooks, process them and send them to Slack or MsTeams channels in a more human readable format, with direct links to the Mist Dashboard pages.  
alerting, script, webhook  
Readme, Github

**OpenAPI**  
openapi  
Mist APIs documentation in OpenAPI 3.0 specification. This is a manually maintained spec which is mainly for documentation purposes.  
Demo, Readme, Github

**PSK**  
Frontend angular, Backend django  
Web UI to manage Mist PPSK. This interface can be used by super admins and network admins to create, delete, edit the PPSKs, send it by email, or display a configuration QRCode to simplify the mobile devices configuration.  
configuration, psk, webapp  
Demo, Readme, Github

# Postman collection

## Working with the REST API

- Must have
- Keep an eye out for updates!



The screenshot shows the Postman application interface. On the left, the sidebar displays the 'MIST\_TEST\_WORKSPACE\_V01' workspace with various collections, environments, and monitors. The main area shows a POST request for 'https://{{host}}/api/v1/login'. The 'Body' tab is selected, showing a JSON payload:

```
1 curl --location --globoff 'https://{{host}}\api/v1/login' \
2 --header 'Content-Type: application/json' \
3 --data '{'
4   "email": "{{email}}",
5   "password": "{{password}}"
6 }'
```

The response status is 200 OK, and the response body is displayed in the 'Pretty' tab:

```
1 {
2   "email": "test@mistsys.com",
3   "privileges": null,
4   "two_factor_required": true,
5   "two_factor_passed": false
6 }
```

# MISTAPI

## Python API

- Just install it and play with it
- Jupyter Notebooks are great for this
- Shout out to Junipers Thomas Munzer! 



```
>>> apisession.login()
```

```
----- Mist Cloud Selection -----
```

```
0) Europe 01 (host: api.eu.mist.com)
1) Global 01 (host: api.mist.com)
2) Global 02 (host: api.gc1.mist.com)
3) Global 03 (host: api.ac2.mist.com)
4) Global 04 (host: api.gc2.mist.com)
```

```
Select a Cloud (0 to 5, or q to exit):
```

```
>>> apisession.login()
```

```
----- Login/Pwd authentication -----
```

```
Login: tmunzer@juniper.net
```

```
Password:
```

```
[ INFO ] Authentication successful!
```

```
Two Factor Authentication code required: 122749
```

```
[ INFO ] 2FA authentication successed
```

```
----- Authenticated -----
```

```
Welcome Thomas Munzer!
```

# Webhook testing

## MIST specific

- Watch out: Auto configures your org to send Webhooks to the tool
- Use it when you are integrating with your monitoring/alerting tools



The screenshot shows a web-based interface for managing webhooks. At the top, there's a header with the Juniper Mist AI logo and the URL "webhook.mist-lab". Below the header is a search bar labeled "Filter" and a table with two columns: "Date" and "TOPIC". Three rows of data are listed:

Date	TOPIC
11/04/23, 09:43:34	audits
11/04/23, 09:43:08	audits
11/04/23, 09:43:08	audits

On the right side of the interface, there are two tabs: "Event" and "Full Message". The "Full Message" tab is selected, displaying a JSON object representing the webhook message. The JSON content is as follows:

```
{  
  "topic": "audits",  
  "events": [  
    {  
      "admin_name": "Steinn Bjarnarson steinnb@advania.is",  
      "id": "6e113cab-227f-45ad-bb28-111111111111",  
      "message": "Accessed Org \"POC-ORG-ADVANIA\"",  
      "org_id": "06e859b6-1827-4122-adf6-111111111111",  
      "src_ip": "15.237.122.52",  
      "timestamp": 1681206188.69912,  
      "user_agent": "unknown_user_agent"  
    }  
  ]  
}
```

# Webhook testing 2

## Generic option

- This is a generic option for any Webhook service
- No auto-configuration of your org



The screenshot shows the Webhook.site interface. At the top, there is a navigation bar with links for 'Webhook.site', 'Docs & API', 'Custom Actions', 'WebhookScript', 'Terms & Privacy', and 'Support'. On the right side of the header are buttons for 'Copy', 'Edit', 'New', 'Login', and 'Upgrade'.

The main content area has a heading 'REQUESTS (0/500) Newest First' and a search bar labeled 'Search Query'. Below this, a message says 'Waiting for first request...'. To the right, there is a section titled 'Your unique URL (Please copy it from here, *not* from the address bar!)' with a URL: <https://webhook.site/c816ce72-36cb-4012-ad42-c6e8c02a5aa2>. Below the URL are 'Copy to clipboard' and 'Open in new tab' buttons.

Further down, there is a section for 'Your unique email address' with an email address: [c816ce72-36cb-4012-ad42-c6e8c02a5aa2@email.webhook.site](mailto:c816ce72-36cb-4012-ad42-c6e8c02a5aa2@email.webhook.site). It also includes 'Copy to clipboard' and 'Open in mail client' buttons.

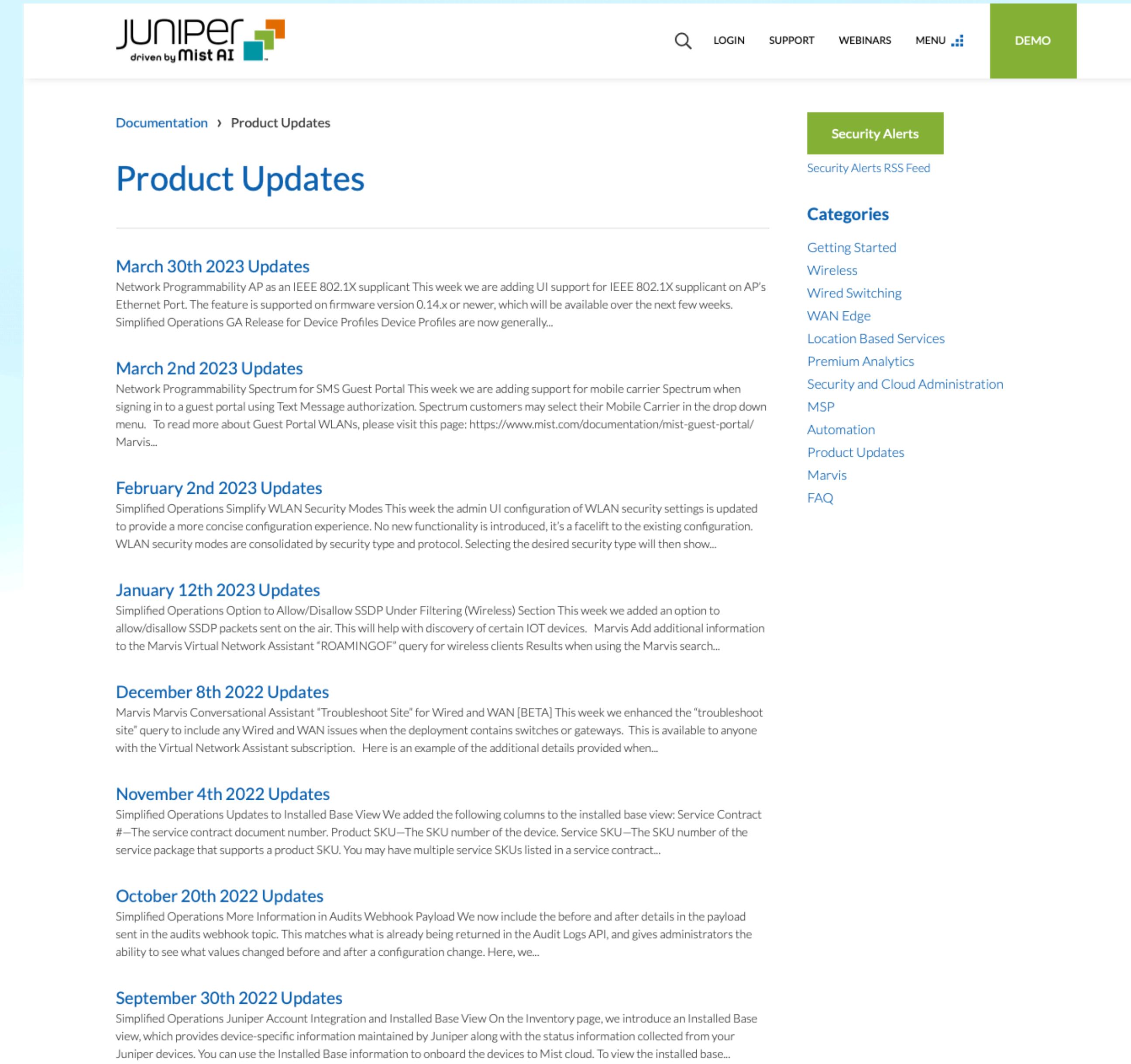
At the bottom of the main content area, there is a note: 'Are you not receiving anything? Make sure that you copied the URL from above, and *not* from the browsers' address bar.' Below this, there is a link to 'Star on GitHub'.

At the very bottom, there is a table with sections for 'Request Details', 'Permalink', 'Raw content', and 'Headers'. Under 'Request Details', there are fields for 'Date', 'Size' (0 bytes), and 'ID'. Under 'Query strings', it says '(empty)'. Under 'Form values', it also says '(empty)'.

# Mist update monitoring

f5, f5, f5, f5

- Mist team is agile and fast, too much so at times...
- The product update site has no RSS / email update options



The screenshot shows the Juniper Mist AI Product Updates page. At the top right, there are links for LOGIN, SUPPORT, WEBINARS, MENU, and DEMO. Below the menu, there's a green button labeled "Security Alerts" and a link to "Security Alerts RSS Feed". On the left, there's a sidebar titled "Categories" with links to various topics like Getting Started, Wireless, Wired Switching, WAN Edge, Location Based Services, Premium Analytics, Security and Cloud Administration, MSP, Automation, Product Updates, Marvis, and FAQ. The main content area displays a series of update entries:

- March 30th 2023 Updates**  
Network Programmability AP as an IEEE 802.1X supplicant This week we are adding UI support for IEEE 802.1X supplicant on AP's Ethernet Port. The feature is supported on firmware version 0.14.x or newer, which will be available over the next few weeks. Simplified Operations GA Release for Device Profiles Device Profiles are now generally...
- March 2nd 2023 Updates**  
Network Programmability Spectrum for SMS Guest Portal This week we are adding support for mobile carrier Spectrum when signing in to a guest portal using Text Message authorization. Spectrum customers may select their Mobile Carrier in the drop down menu. To read more about Guest Portal WLANs, please visit this page: <https://www.mist.com/documentation/mist-guest-portal/> Marvis...
- February 2nd 2023 Updates**  
Simplified Operations Simplify WLAN Security Modes This week the admin UI configuration of WLAN security settings is updated to provide a more concise configuration experience. No new functionality is introduced, it's a facelift to the existing configuration. WLAN security modes are consolidated by security type and protocol. Selecting the desired security type will then show...
- January 12th 2023 Updates**  
Simplified Operations Option to Allow/Disallow SSDP Under Filtering (Wireless) Section This week we added an option to allow/disallow SSDP packets sent on the air. This will help with discovery of certain IOT devices. Marvis Add additional information to the Marvis Virtual Network Assistant "ROAMINGOF" query for wireless clients Results when using the Marvis search...
- December 8th 2022 Updates**  
Marvis Marvis Conversational Assistant "Troubleshoot Site" for Wired and WAN [BETA] This week we enhanced the "troubleshoot site" query to include any Wired and WAN issues when the deployment contains switches or gateways. This is available to anyone with the Virtual Network Assistant subscription. Here is an example of the additional details provided when...
- November 4th 2022 Updates**  
Simplified Operations Updates to Installed Base View We added the following columns to the installed base view: Service Contract #—The service contract document number. Product SKU—The SKU number of the device. Service SKU—The SKU number of the service package that supports a product SKU. You may have multiple service SKUs listed in a service contract...
- October 20th 2022 Updates**  
Simplified Operations More Information in Audits Webhook Payload We now include the before and after details in the payload sent in the audits webhook topic. This matches what is already being returned in the Audit Logs API, and gives administrators the ability to see what values changed before and after a configuration change. Here, we...
- September 30th 2022 Updates**  
Simplified Operations Juniper Account Integration and Installed Base View On the Inventory page, we introduce an Installed Base view, which provides device-specific information maintained by Juniper along with the status information collected from your Juniper devices. You can use the Installed Base information to onboard the devices to Mist cloud. To view the installed base...

# Mist update monitoring using Visualping

- Make your own!
- “free” 1x a day monitoring of any website
- Visual / Text / HTML element change monitoring



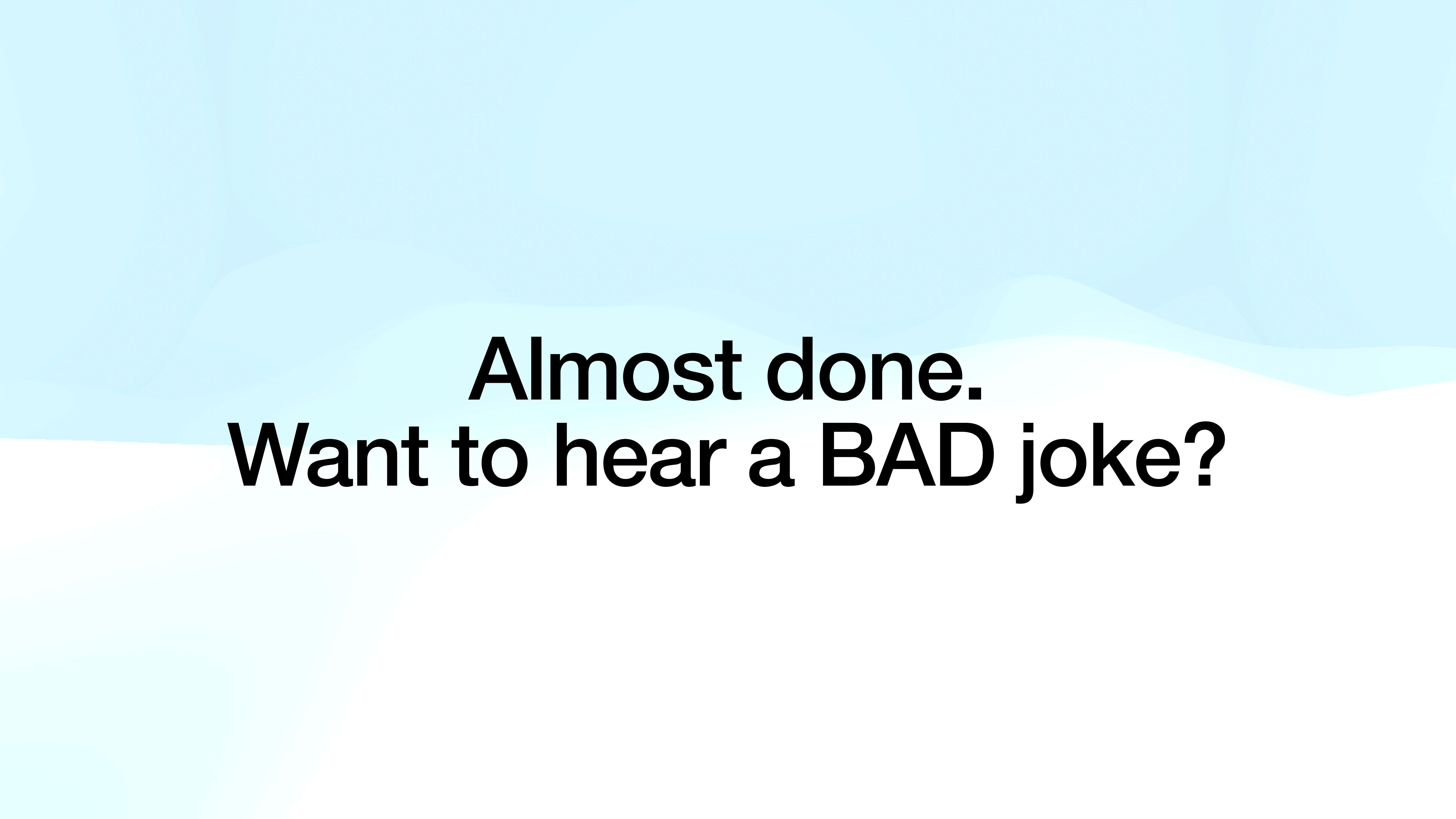
https://www.mist.com/documentation/category/product-updates/

The screenshot shows the Mist documentation page for product updates. It displays three recent updates: 'March 30th 2023 Updates', 'March 2nd 2023 Updates', and 'February 2nd 2023 Updates'. Each update entry includes a brief description and a link to the full update page. On the right side, there is a sidebar with categories like Configuration, Automation with APIs, and Network Assurance. Below the sidebar, there are fields for sending change alerts (to me@example.com) and setting the monitoring frequency (Every Day). At the bottom, there is a prominent blue button labeled 'Start free monitoring'.

The screenshot shows the Visualping dashboard interface. It displays a comparison between two versions of the Juniper MIST feature updates page. A green checkmark indicates a visual difference was detected. The dashboard includes a summary of the changes, download links for the old and new versions, and a message from Visualping stating they detected a medium change on the Juniper MIST feature updates page. At the bottom, there is a link to the March 30th 2023 Updates page.

# Inclusion

- It's never been easier to get started
- Tooling is stabilizing
- We can become even more ambitious as “AI” tools become more powerful.



**Almost done.  
Want to hear a BAD joke?**

**“Why did the IT team throw a party when they deployed Juniper MIST?**

**Because they knew they were in for an  
AI-mazing time!”**

**ChatGPT 2023**

**Thanks for listening**  
**Q&A**



**SLIDES**



**One more thing!**

# **Switch rip and replace, example**

# Brownfield customer switch replacement

## One way to do it.

- Gather information about the existing switch
  - Port counts
  - PoE budget etc.
- Go and configure line-by-line matching VLAN/Port assignments on the new switch
- Rack and carefully move all the connections from the old switch to the new switch

**Another way  
to do this....**

# **Warning**

**This is a bad idea!**



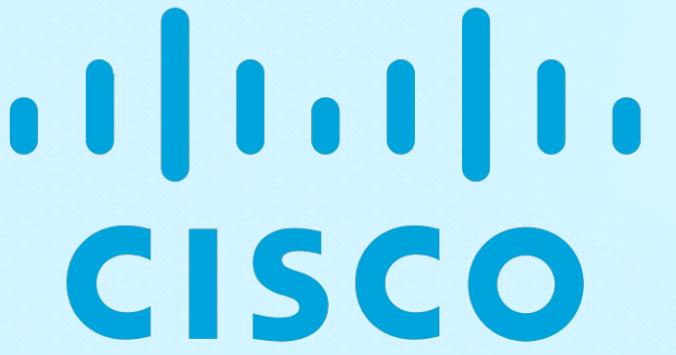
Can you give me a list of OUI to VLAN mappings?



Here you go!



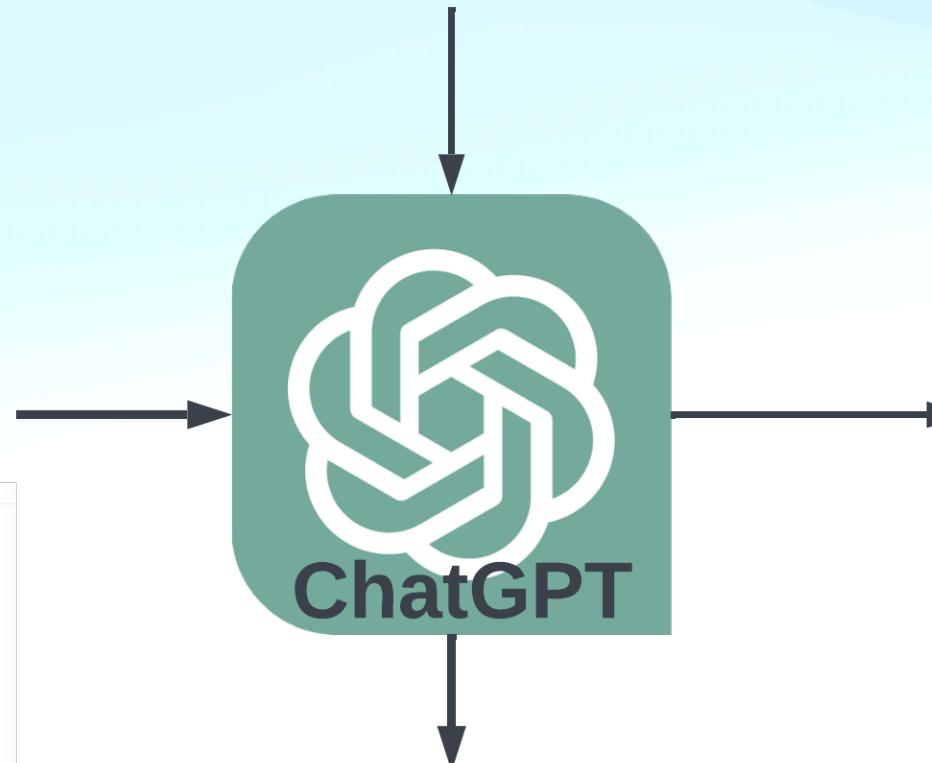
VLAN	OUI (First 6 characters)	Vendor
84	001871	Hewlett Packard Enterprise
	70b3d5	Dell Inc.
	001b4f	Avaya
85	70b3d5	Dell Inc.
	1c1d86	Cisco Systems, Inc.
	001b4f	Avaya
101	3cfdfc	Intel Corporate
230	2c3033	NETGEAR
	00e0fc	HPE
	001b4f	Avaya
243	001871	Hewlett Packard Enterprise
	00e0fc	HPE



```
router01#show arp dynamic
Protocol Address Age (min) Hardware Addr Type Interface
Internet 172.18.120.13 - a493.4c22.1111 ARPA GigabitEthernet0/1.3006
Internet 172.18.120.14 174 d46d.5025.2222 ARPA GigabitEthernet0/1.3006
Internet 172.18.120.62 2 a493.4c22.3333 ARPA GigabitEthernet0/0.84
Internet 172.18.120.222 0 a493.4c22.4444 ARPA GigabitEthernet0/0.85
Internet 172.18.124.21 96 3c08.f617.5555 ARPA GigabitEthernet0/0.101
```



```
1 curl --location --request PUT 'https://api.eu.mist.com/
  api/v1/orgs/nononono/networktemplates/
  96400f12-87ff-4975-a8ae-f80b84558abc' \
2 --header 'Accept: application/json' \
3 --header 'Content-Type: application/json' \
4 --header 'Authorization: Token nononono' \
5 --header 'Cookie: csrf-token.
eu=IM1Swy5BL9TGWPjUXhg9qCDZxvn552rqzwDqfX6ypS7RzXq
doQfY8k5jM0t26mx' \
6 --data '{
7   "port_usages": {
8     "dynamic": {
9       "mode": "dynamic",
10      "rules": [
11        {
12          "src": "link_peermac",
13          "usage": "guest",
14          "equals": "00:18:71",
15          "expression": "[0:8]"
16        }
17      ]
18    }
19  }
20}'
```



VLAN	OUI (First 6 characters)	Vendor
84	001871	Hewlett Packard Enterprise
	70b3d5	Dell Inc.
	001b4f	Avaya
85	70b3d5	Dell Inc.
	1c1d86	Cisco Systems, Inc.
	001b4f	Avaya
101	3cfdfc	Intel Corporate
230	2c3033	NETGEAR
	00e0fc	HPE
	001b4f	Avaya
243	001871	Hewlett Packard Enterprise
	00e0fc	HPE

Home Workspaces API Network Explore Search Postman Invite Settings Notifications Upgrade

MIST\_TEST\_WORKSPACE\_V01 New Import Overview Mist Cloud GET getOrgNetw GET getOrgDe PUT updateOr advania poc + No Environment

Collections APIs Environments Mock Servers Monitors Flows History

+ / O.. Netw... / updateO... Save ...

PUT https://{{host}}/api/v1/orgs/:org\_id/networkitem Send

Params Auth Headers (14) Body Pre-req. Tests Settings

raw JSON Beautify

1 {  
2     "port\_usages": [  
3         "dynamic": {  
4             "mode": "dynamic",  
5             "rules": [  
6                 {  
7                     "src": "link\_peermac",  
8                     "usage": "guest",  
9                     >equals": "00:18:71",  
10                     "expression": "[0:8]"  
11                 },  
12                 {  
13                     "src": "lldp\_system\_description",  
14                     "usage": "uplink",  
15                     >equals": "Juniper",  
16                     "expression": "[0:7]"  
17                 },  
18                 {  
19                     "src": "lldp\_system\_description",  
20                     "usage": "ap",  
21                     >equals": "Mist",  
22                     "expression": "[0:4]"  
23                 }  
24         }  
25     }  
26     }  
27 }

cURL

```
curl --location --request PUT 'https://api.eu.mist.com/api/v1/orgs/nononono/networktemplates/'  
--header 'Accept: application/json'  
--header 'Content-Type: application/json'  
--header 'Authorization: Token nononono'  
--header 'Cookie: csrftoken.'
```

--data '{  
    "port\_usages": [  
        "dynamic": {  
            "mode": "dynamic",  
            "rules": [  
                {  
                    "src": "link\_peermac",  
                    "usage": "guest",  
                    >equals": "00:18:71",  
                    "expression": "[0:8]"  
                },  
                {  
                    "src": "lldp\_system\_description",  
                    "usage": "uplink",  
                    >equals": "Juniper",  
                    "expression": "[0:7]"  
                },  
                {  
                    "src": "lldp\_system\_description",  
                    "usage": "ap",  
                    >equals": "Mist",  
                    "expression": "[0:4]"  
                }  
            }  
        }  
    }  
}

can you make mappings based on this table and make rules based on each mac OUI, i've already added the 00:18:71

```
"dynamic": {  
    "mode": "dynamic",  
    "rules": [  
        {  
            "src": "link_peermac",  
            "usage": "84",  
            "equals": "00:18:71",  
            "expression": "[0:8]"  
        },  
    ]  
}
```



Certainly! Here are the rules based on the table provided, excluding the rule for "00:18:71" which you have already added:

```
css  
  
"dynamic": {  
    "mode": "dynamic",  
    "rules": [  
        {  
            "src": "link_peermac",  
            "usage": "84",  
            "equals": "00:18:71",  
            "expression": "[0:8]"  
        },  
        {  
            "src": "link_peermac",  
            "usage": "84",  
        }  
    ]  
}
```



Here is the updated CURL call with the new rules:

bash

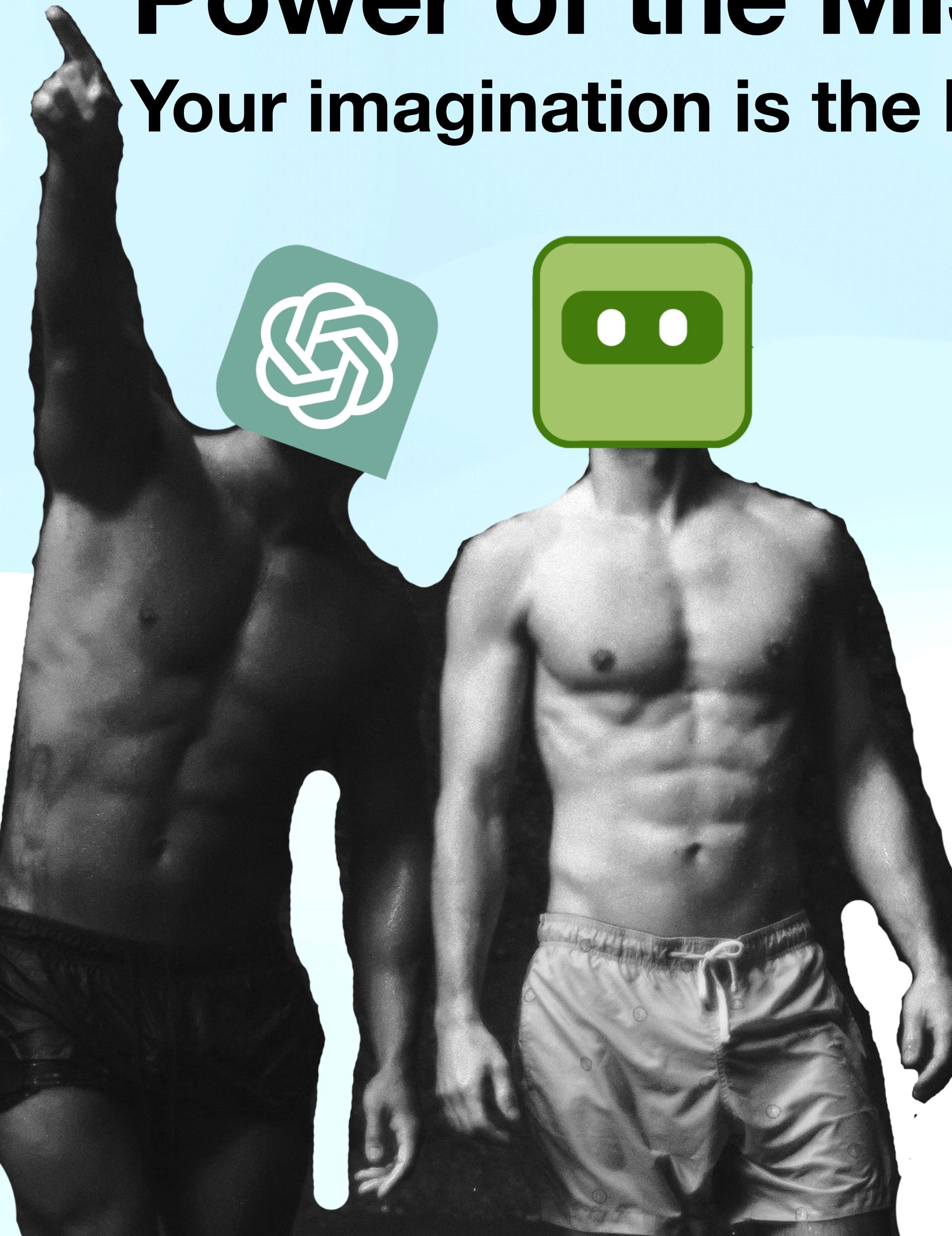
Copy code

```
curl --location --request PUT 'https://api.eu.mist.com/api/v1/orgs/nononono/'  
--header 'Accept: application/json' \  
--header 'Content-Type: application/json' \  
--header 'Authorization: Token nononono' \  
--header 'Cookie: csrftoken.eu='  
--data '{  
    "port_usages": {  
        "dynamic": {  
            "mode": "dynamic",  
            "rules": [  
                {  
                    "src": "link_peermac",  
                    "usage": "84",  
                    "equals": "00:18:71",  
                    "expression": "[0:8]"  
                },  
                {  
                    "src": "link_peermac",  
                    "usage": "84",  
                    "equals": "70:B3:D5",  
                    "expression": "[0:8]"  
                },  
                {  
                    "src": "link_peermac",  
                    "usage": "84",  
                    "equals": "00:1B:4F",  
                    "expression": "[0:8]"  
                }  
            ]  
        }  
    }  
}'
```

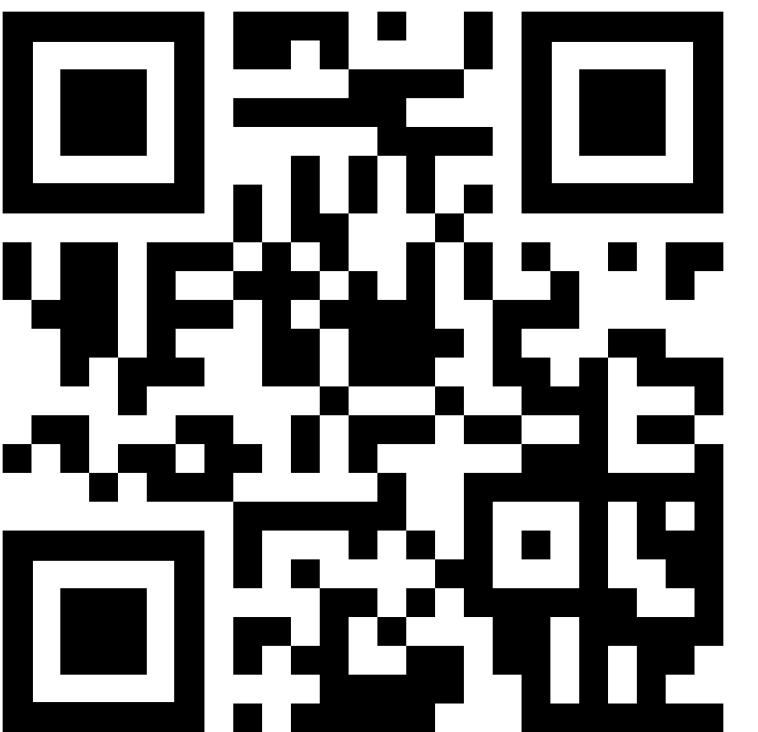
# Power of the MIST API

Your imagination is the limit.

- We took a manual task
- Used an LLM to parse the data
- Invoked the power of the Mist API to do exactly what we wanted.
- Dynamic port configuration is doing the heavy lifting here.
- DON'T TRUST THE LLM LIKE YOU TRUST MARVIS!



# Thanks for listening!



SLIDES

<https://github.com/steinzi>